Shared Flow

Table of contents

1. Introduction

A shared flow (a kind of controller class) provides a place for actions, exception handlers and data that the developer wants to make available to multiple page flows. You can have multiple shared flows, each with different delegated functions. For example, you might have one shared flow specifically for handling exceptions, and another shared flow to provide common data resources.

Shared flows can reside anywhere in your web app, and can be invoked from other controller files or from JSP files.

2. Creating a Shared Flow

To create a shared flow controller file

1. decorate the class with the @Controller annotation
2. extend the SharedFlowController class

```
import org.apache.beehive.netui.pageflow.annotations.Jpf;
import org.apache.beehive.netui.pageflow.SharedFlowController;

@Jpf.Controller
public class SharedFlow extends SharedFlowController
{

    ...

}
```

3. Adding Actions and Data to a Shared Flow

Shared flows have all of capabilities of other controller files. They can contain

1. ordinary Java methods
2. simple actions (actions defined by the @Jpf.SimpleAction annotation)
3. action methods (methods decorated with the @Jpf.Action annotation)
4. exception handling code (via the @Jpf.Catch annotation)
5. common data
6. and other sorts of functionality...

Suppose you wanted a common handler for the generic `Exception.class` whenever and wherever it is thrown in your web app. To add this sort of exception handling, you would (1) add a @Jpf.Catch annotation and (2) add the appropriate exception handling method to the shared flow. You could, of course, add the same code to each controller file in your application, but that would make for redundant code. By adding the exception handling code to the shared flow, you have a centralized, non-redundant way to handle a given exception

type whenever it arises.

The @Jpf.Catch annotation appears as an attribute of the parent @Jpf.Controller annotation. The @Jpf.Catch annotation added below means, roughly, "When the exception `Exception.class` is thrown, invoke the method `handleGenericException` to handle it."

```
import org.apache.beehive.netui.pageflow.annotations.Jpf;
import org.apache.beehive.netui.pageflow.SharedFlowController;

@Jpf.Controller(
    catches={
            @Jpf.Catch(method="handleGenericException",
type=Exception.class)
    }
)
public class SharedFlow extends SharedFlowController
{

    ...

}
```

The exception handling method appears as a method decorated with the `@Jpf.ExceptionHandler` annotation.

```
import org.apache.beehive.netui.pageflow.annotations.Jpf;
import org.apache.beehive.netui.pageflow.SharedFlowController;

@Jpf.Controller(
    catches={
            @Jpf.Catch(method="handleGenericException",
type=Exception.class)
    }
)
public class SharedFlow extends SharedFlowController
{

    @Jpf.ExceptionHandler(
        forwards={
            @Jpf.Forward(name="errorPage", path="/error.jsp")
        }
    )
    protected Forward handleException(Exception ex, String actionName,
String message, Object form) {
        // ...other error handling code here...
        return new Forward("errorPage");
    }

}
```

4. Invoking Shared Flows from Other Controller Files

To invoke the functionality of shared flow, it must first be declared on the invoking resource. To declared a shared flow on another controller file, use the @Jpf.SharedFlowRef annotation (an attribute of the parent @Jpf.Controller annotation).

```
@Jpf.Contoller(
    sharedFlowsRefs = {
        @Jpf.SharedFlowRef(name = "someNameOne", type =
"SharedFlowClassOne.class"),
        @Jpf.SharedFlowRef(name = "someNameTwo", type =
"SharedFlowClassTwo.class")
    }
)
public class SomeController extends Controller
```

and then create a member of your shared flow class. The member must be decorated with the @Jpf.SharedFlowField annotation.

```
@Jpf.Contoller(
    sharedFlowsRefs = {
        @Jpf.SharedFlowRef(name = "sharedFlowOne", type =
"SharedFlowClassOne.class"),
        @Jpf.SharedFlowRef(name = "sharedFlowTwo", type =
"SharedFlowClassTwo.class")
    }
)
public class SomeController extends Controller
{

    @Jpf.SharedFlowField(name="sharedFlowOne")
    private SharedFlowClassOne _sharedFlowOne = null;

}
```

You will recieve a compiler error if you annotate a member field as a shared flow (with @Jpf.SharedFlowField) without having first declared the shared flow in the Controller annotation (with @Jpf.SharedFlowRef).

Note:

You cannot access actions in `Global.app` if you've declared shared flow references (@Jpf.Controller(sharedFlowRefs=...)) in your page flow.

Now you can invoke the resources within the shared flow. For example, to invoke the shared flow's `handleLogout()` method, do the following:

```
    _sharedFlowOne.handleLogout();
```

5. Invoking Shared Flows from JSP Pages

Shared flow resources can be accessed from a JSP using the scope `sharedFlow`, for example,

```
<netui-data:repeater dataSource="sharedFlow.specificSharedFlow.someData">
```

Actions declared in a SharedFlow are accessed by specifying a fully qualified action

```
<beehive-petstore:catalogNav
action="sharedFlow.specificSharedFlow.someAction"
labelValue="${bundle.view.mainMenuLabel}"/>
```

where `specificSharedFlow` was the name given to the shared flow in the `@Jpf.SharedFlowRef/@Jpf.SharedFlowField` annotations.

```
@Jpf.SharedFlowRef(name = "specficSharedFlow", type =
"SomeSharedFlow.class")

@Jpf.SharedFlowField(name="specificSharedFlow")
```

6. Example Code

See the shared flow controller file in the PetStore sample, located at `<BeehiveRoot>/samples/petstoreWeb/webappRoot/SharedFlow.java`